Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718-1876 Electronic Version
VOL 7 / ISSUE 3 / DECEMBER 2012 / 74-87

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762012000300008

Location-aware Mobile Services for a Smart City: Design, Implementation and Deployment

Luca Calderoni[1], Dario Maio[2] and Paolo Palmieri[3]

Università di Bologna, DISI, Bologna, Italia, [1]luca.calderoni@unibo.it, [2]dario.maio@unibo.it
[3]Université Catholique de Louvain, UCL Crypto Group, Louvain-la-Neuve, Belgium, paolo.palmieri@uclouvain.be

Received 30 April 2012; received in revised form 9 August 2012; accepted 9 August 2012

Abstract

A *smart city* is a high-performance urban context, where citizens are more aware of, and more integrated into the city life, thanks to an intelligent city information system. In this paper we design, implement and deploy a smart application that retrieves and conveys to the user relevant information on the user's surroundings. This case study application let us discuss the challenges involved in creating a location-aware mobile service based on live information coming from the city IT infrastructure. The service, that is currently being deployed in the Italian city of Cesena, has been designed with the goal of being a general model for future applications. In particular, we discuss location-aware and mobile development, cloud and cluster based geographical data storage, and spatial data computation. For each of these topics we provide implementation and deployment solutions based on currently available technology. In particular we propose an architecture based on a complex On-Line Transaction Processing (OLTP) infrastructure. Furthermore, this paper represents the first comprehensive, scientific study on the subject.

Keywords: Smart city, Android, MySQL cluster, Location-aware applications, Spatial indices, Range query

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718-1876 Electronic Version
VOL 7 / ISSUE 3 / DECEMBER 2012 / 74-87

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762012000300008

1 Introduction

Smart cities are a lively and growing research topic. In recent years, several scientific studies on the subject have been written, presented and discussed. A comprehensive overview of the state of the art in this research field is available in [23]. While some research works focus on a high-level vision of the urban context, and aim at giving us an idea of how a smart future city could appear to the citizen's eyes, others describe a technological model or service for the city, as we do in this paper.

Over the years many different definitions of *smart city* have been proposed. Nam and Pardo, analyzing a number of these definitions in two closely related works [18]-[19], notice that most works can be classified into different categories as they discuss future smart cities adopting either an architectonic, social or infrastructural point of view, or a combination of the three. However, independently from the chosen point of view, every contribution aims at urban innovation: in the first case trying to design the exterior and tangible part (architecture, ecology, etc.), in the second the relational part (governance, policy and citizen interaction) and in the third by focusing on the infrastructures, mainly technological and internet based, that combine and connect the city intelligent systems. Depending on the approach used, each work adopts the definition that best fits the specific context.

Another theme that is central to the smart city concept and recurs in its definitions is that of citizen interaction. This interaction transforms the citizen from a passive subject into a live actor, integral part of the system [5], [24], which in turn becomes a real, living ecosystem, sometimes called *living lab* [6].

In ICT, the concept of smart city expands that of intelligent and integrated networks such as the *ubiquitous sensor networks* (USN) [13]. In fact, a smart city receives and integrates knowledge from the Internet of Things (IoT), the Internet of Services (IoS) and the Internet of People (IoP) [10], [12]. For instance, the intelligent integration of data coming from traffic sensors, weather stations and security cameras spread around the city could be used to provide users (such as citizens, ambulances, taxis) with real-time traffic information and route selection based on the current state of the urban route network. Actors of city automation do not generally need to be human. In fact, another interaction common of smart cities is that of city-to-enterprise infrastructure (as opposed to city-to-citizen) [14]. For example, real-time monitoring of the city electrical grid can be used to automatically warn industrial control systems of possible criticalities in the power supply.

In [10], Hernández-Muñoz et al. discuss the issues related to the deployment of massive sensor networks and the definition of reusable services for citizens and administrations. In this research work, the urban environment is also seen as an open innovation platform to perform large scale experimentation, a concept similar to the one of *Living Lab* proposed in [24]. In [21], Park et al. describe a cloud computing model offering various kind of IT services for the citizen. The management and monitoring of the cloud itself is performed using mobile Android OS clients.

The possibility of collecting location- and time-aware data from the urban context is a particularly interesting feature of a smart city. Those data are in fact fundamental in order to offer high value services to the citizen [2], and they can even serve the purpose of analyzing and understanding the inner dynamics and functioning of the city - social, economical and even psychological [22]. For instance, location- and time-aware data have been used for urban security, usually involving video control and urban safety applications, and for traffic monitoring and optimal route finding, often associated with automated recognition of road signage and traffic sensors [1], [27].

In this paper we discuss the design, implementation and deployment of a mobile location-aware application for a smart city. While we present a specific application as a case study, the general aim of this work is to discuss the challenges presented by the creation of such an application, with specific focus on the information technology point of view. For the purpose of this work, we define a smart city as a high-performance urban context, where citizens are interconnected to each others and to the city itself, which provides a constant flow of information, personalized to the user's needs and preferences. Citizens are more independent, more aware of the surrounding opportunities, and benefit from the integrated services that the city offers. The ICT infrastructure is the brain of the city, governs its body and reacts to the circumstances intelligently. A smart city allows new ideas to prosper and new, more efficient approaches to be developed in economy, politics, governance, mobility, environment and all the other aspects of city life. Following this definition, we model the functioning of a smart city in a three-layers scheme, as shown in Figure 1. The three-levels ecosystem gathers data from the world (sensor networks, world wide web and user contribution), stores and processes these data into a kernel (the city brain), and exposes through several services the processed data. These services can be designed to be used by citizens, but also by non-human actors (for instance, in the case of direct communication with industry automated subsystems). At the *access layer*, we have the sources of information for a smart city: sensor networks, relevant knowledge bases made available over the Internet, and the citizens themselves through user contribution. This information is processed at the *kernel layer*, where raw data are gathered and then elaborated by the computing infrastructure (*kernel*) of the city, the city brain. Citizens, and other relevant city actors (such as businesses, industries, local government) query the system and access information through dedicated services and interfaces (the *service layer*).

The case study we discuss in this paper is an *around me* application, that provides users on the go with information about their current location and its surroundings. The information collected by the system and served to the users is a catalog of interesting objects located within the city. By "interesting objects" we mean shops and businesses, touristic landmarks or useful public services such as hospitals and pharmacies. However, depending on the application, also traffic sensors, weather stations, wireless hotspots, or any other type of point of interest (PoI) can be included. Each object is classified into a specific category, and its geographical location and other relevant information are stored. The user performs searches through a location-aware mobile application, that displays search results as a list of objects, chosen according to the user's position and preferences. The system we propose integrates information coming from pre-existing knowledge bases and user generated content. In fact, a user can suggest interesting objects and shape

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718-1876 Electronic Version
VOL 7 / ISSUE 3 / DECEMBER 2012 / 74-87

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762012000300008

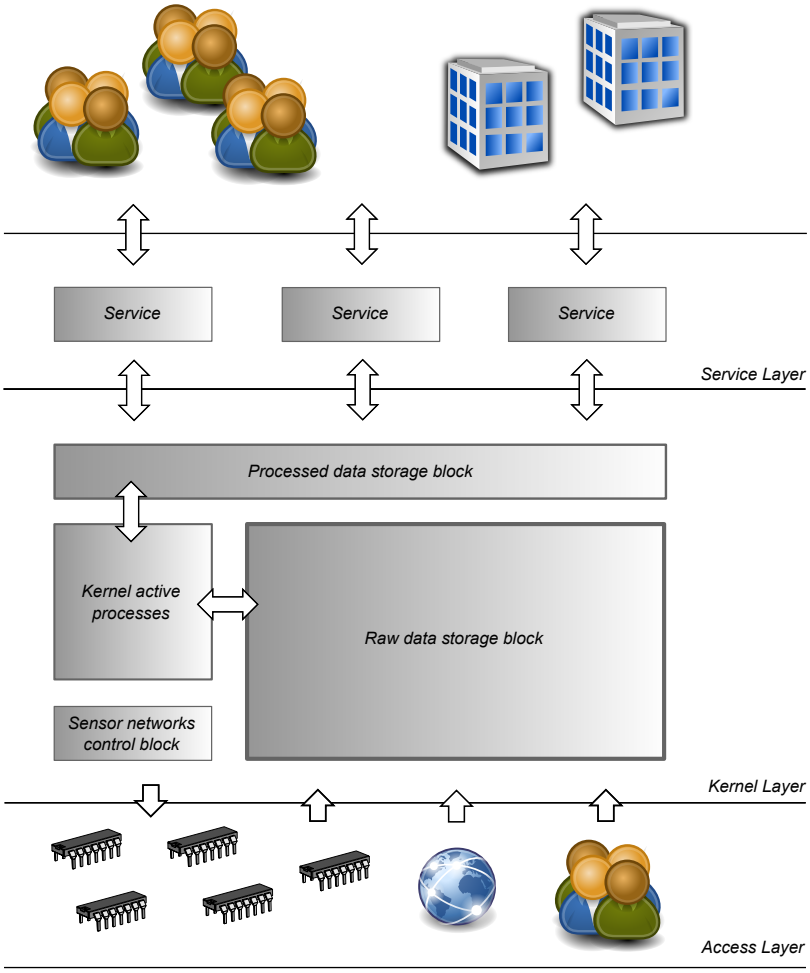


Figure 1: A schematic high-level representation of the underlying infrastructure of a future smart city.

the system around her own needs.

The kernel of our system is a distributed computing infrastructure based on a clustered database management system (DBMS). The external interface is provided by a web service, that users can query though a specific software developed for mobile phones. We choose this platform in order to offer a high degree of flexibility to users on the move, that are the typical target of this kind of application. Moreover, modern mobile phones (correctly called smartphones), are equipped with all the tools needed to realize a location-aware application, such as a GPS receiver and Internet connection.

A first experimental implementation of the service is currently being deployed in Cesena (Italy). Results from this experiment will be used to improve the framework of the service for future releases.

1.1 Smart Cities Today

Smart cities are close to becoming a common reality. Recently, many experiments and pioneering projects have been successfully run. Some of these works cover specific applications and services suitable for smart cities (often involving mobile devices), while other are integrated plans for an entire city, both theoretical or actually implemented. An example of the latter is the plan for the construction of an entire city outside Abu Dhabi: *Masdar City* [9]. The city will cover an area of 6 square kilometers and be home to about 50.000 citizens. A major feature of the project is an integrated public transport system, that will minimize the need of private vehicles.

In Zaragoza, Spain, a wide sensor network controls approximately 90% of urban routes. A centralized control point monitors in real-time the traffic of the whole city, and helps the government to improve the road network with adequate policies [9].

Another Spanish city, Santander, is home to one of the most ambitious experiments, that transformed the city into a real living laboratory. *Smart Santander* [9]-[10] represents a true *Internet of Things* case study [12], housing thousands of devices that form a number of sensor networks. The project collects data from the urban environment and makes the information gathered available to research groups around the world. The aim of this project is to help design experimental services like traffic and parking remote control, tourism guidance through mobile devices or video monitoring

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718-1876 Electronic Version
VOL 7 / ISSUE 3 / DECEMBER 2012 / 74-87

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762012000300008

of sensible areas.

The European Union is particularly active in supporting the development of medium-sized cities, contrary to current research that tends to focus on the *global* metropolises. Through the ranking of 70 European cities, the EU aims at sparking competition and therefore help optimize the internal organization and resource allocation practices of those cities and, ultimately, improve living standards [7].

2 Design and Architecture

The mobile application we present as a case study helps users (citizens) to locate useful objects around their current location. The information is retrieved by querying the server infrastructure, and is presented ordered by distance from the user. Distances are updated during the user's movements. In the design of the application and of the computing infrastructure behind it, we adopt a pragmatic, practical approach, based on the integration of existing technologies and newly proposed solutions.

Let us imagine a person visiting a city for the first time. She has no idea of what the city has to offer, but she is interested in finding out about museums. With a mobile device on her (phone or tablet) and an Internet connection, she can use this application to retrieve in a few seconds all the surrounding museums. For each of them, she can than request additional details like the entrance cost, opening hours and so on. But the application is not limited to touristic information. Instead, it can display a large variety of objects, to best satisfy the user's needs in every context. For instance, a collection of pharmacies, hospitals and medical centres is useful even for citizens in their own city.

In the design of our solution, however, we do not restrict ourselves to a specific city or geographical region. In fact, the provider side that serves the client application is designed in order to be fully scalable, and to be able to serve requests that range from a single city to all over the world.

In our test case application we consider a number of well-known categories of common urban objects, such as book-stores, cinemas, gas stations, supermarkets and so on. For each category, the provider of the service retrieves relevant information for all of the objects around the world, geo-locates them and fills the database with this information. If we consider an instance of the application covering a small city, objects could be manually entered by the staff. If instead we consider a wider context, an automated solution is needed. Object information and addresses can be crawled from the world wide web, for example considering only well-known address directories in order to find consistent results.

User involvement is a fundamental component of applications designed for a smart urban context [2], [6], [23]-[24]. In our application users are encouraged to submit new objects and categories, or improvements and corrections to the existing ones. For instance, a user could tag a well-formed object in a web page, telling the crawler to consider it when searching for addresses. We also consider a community of citizens that insert or tag objects in many ways, for example from a web browser interface or directly from the mobile application. A reward system could prompt users to participate.

Our solution is based on a client-server architecture, where the clients are mobile devices that interact with a distributed server infrastructure. We outline the structure of a dedicated mobile application and the corresponding remote interface in Section 2.2, where we provide design specifications and the complete components diagram, and we discuss the client-server architecture. We design the provider side with performance, scalability and geographical distribution in mind. Since we aim at serving requests coming from all over the world, the provider-side infrastructure has to be able to serve billions of requests per day and perform queries on massive amounts of data. Deployment and implementation are discussed in depth in the following, in Section 3.

2.1 Stakeholder Analysis

Given the social relevance of any project that is developed for a city and that will shape its future life, an analysis of the stakeholders is particularly important. Table **??** shows the different actors involved in the project, the phases of the project development (project outline, design, implementation, operation, maintenance) in which each of them will take part and their interests in pursuing the project.

Like most smart city innovations, our system is designed to help improve the citizens fruition of the city. Therefore, the local government will benefit from an increased citizen satisfaction, and a better attractiveness to tourism. Tourists themselves, like the inhabitants of the city, will be the final users of the service. The users will be also prompted to directly participate in the project, by suggesting objects or categories to be included in the application. A reward scheme (even monetary) can be put in place to stimulate citizens engagement. The service provider, that is, the entity that will actually develop the system and take care of its maintenance once in operation, will act in close coordination with the local government, especially in the design phase, in order to identify the specific needs of the city. The project could also be run internally by a dedicated team part of the local government. The local government will also help attract external funding originating from private and public investors and innovation initiatives (see Figure 2). A business that is listed in the application database will profit indirectly from the application, thanks to the increased visibility to potential customers. The economic viability of the project is assured by the diversified source of income for the service provider: government participation in the initial funding and operation expenses and sponsoring of the application from the listed business. Advertisement revenues can be collected from businesses that wants to improve their placement in the results proposed by the application to the user.

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718-1876 Electronic Version
VOL 7 / ISSUE 3 / DECEMBER 2012 / 74-87

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762012000300008

Table 1: Stakeholder analysis.

Actor	Involvement	Benefit
Local Government	Opportunity/needs assessment, concept proposal	1. Increased citizen satisfaction 2. Attracting innovation investments
Citizens	User contribution during operation phase	1. Better urban experience 2. Earning opportunities due to user participation
Businesses	Sponsoring during implementation and operation phase	1. Increased business due to wide information diffusion 2. Better customer experience thanks to location-aware advertising
Tourists	Service end users	1. Better urban experience
Service provider	System design and development, service maintenance	1. Strong business enforced by diverse sources of income (government, businesses, other advertisers)

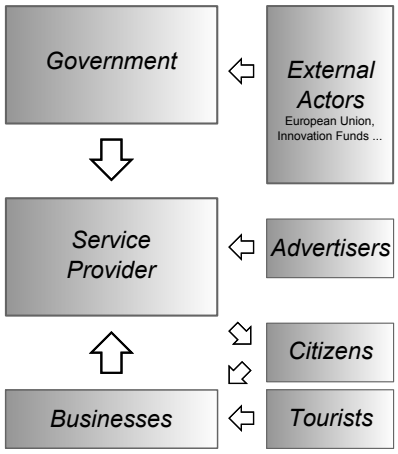


Figure 2: Cash-flow diagram.

2.2 Software Engineering

In this section we provide a detailed scheme representing the architecture of the entire application, designed according to UML (Unified Modeling Language) standards. UML is a visual language for system design and engineering and for object oriented software modeling. In the UML component diagram shown in Figure 3 we can see that the client side is designed as a mobile application for the Android operating system while the server side provides contents to mobile clients through a web service. The client side requires an interface, which is exposed by the provider side. We develop this interface as a web service that serves each client request after authentication.

The client consists in an application for mobile phones. The user interface of the application is composed of two main views. The first handles the objects list, while the second shows the details of a single object. The object list displays, for each object, a short name and the distance from the user. Object categories are identifiable thanks to a representative icon. A user can define custom settings for the application through a settings view. In particular, he can specify the username and password needed to access the service, as well as select from a list the desired object categories to retrieve and the action range that should be used when the application looks for objects.

The server side handles clients requests via an HTTP interface. After the client is recognized (via the authentication module) the server process the request on the database server. Then, the response is delivered to the mobile client, again through HTTP.

Responsiveness is an important factor in real-time services, so we want to serve each request from an application server near the client. Another requirement of the system is scalability of the server side: thousands of real time requests among millions of objects need a very powerful OLTP architecture in order to be quickly served. Finally, a database with as many records as described and a complex data distribution should present some solution for enforcing data availability and to ensure data recovery in case of a server failure. Therefore, we design our server side as a geographically distributed cluster architecture with data replication.

3 Infrastructure

While in the previous sections we discussed the high-level design of our construction, in this part of the paper we discuss the infrastructure. In particular, we present and motivate our choices for the target mobile platform and the DBMS used. We also discuss the cloud framework that will be the base for the provider side, and present a flexible and scalable solution for a geographically distributed service. Scaling a database as a service (DBaaS) to serve large amount of requests is a crucial objective, as explained in [4]. In order to obtain a high-performance OLTP infrastructure we deploy our provider side on a cluster.

The server side deployment and implementation are provided in sections 3.2 and 3.3. In this complex scenario many practical, scientific and technical problems occur. For example, we deal with the problem of efficiently selecting spatial objects around the user from an enormous amount of data. In previous works on the subject, such as [15] and [17], the area surrounding a geo-located point is first treated as a square having the half-side equal to the search radius. This square is identified by a minimum and maximum longitude and a minimum and maximum latitude as well and is often called the *bounding box*. We propose a novel technique for range query optimization, with improvements in the bounding box calculation in Section 3.3.2. In Section 3.3.3 we present results from an experiment we run on database table indices for spatial data. Finally, in Section 3.4 we discuss security and privacy issues for a distributed database and the user application, and propose possible solutions.

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718-1876 Electronic Version
VOL 7 / ISSUE 3 / DECEMBER 2012 / 74-87

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762012000300008

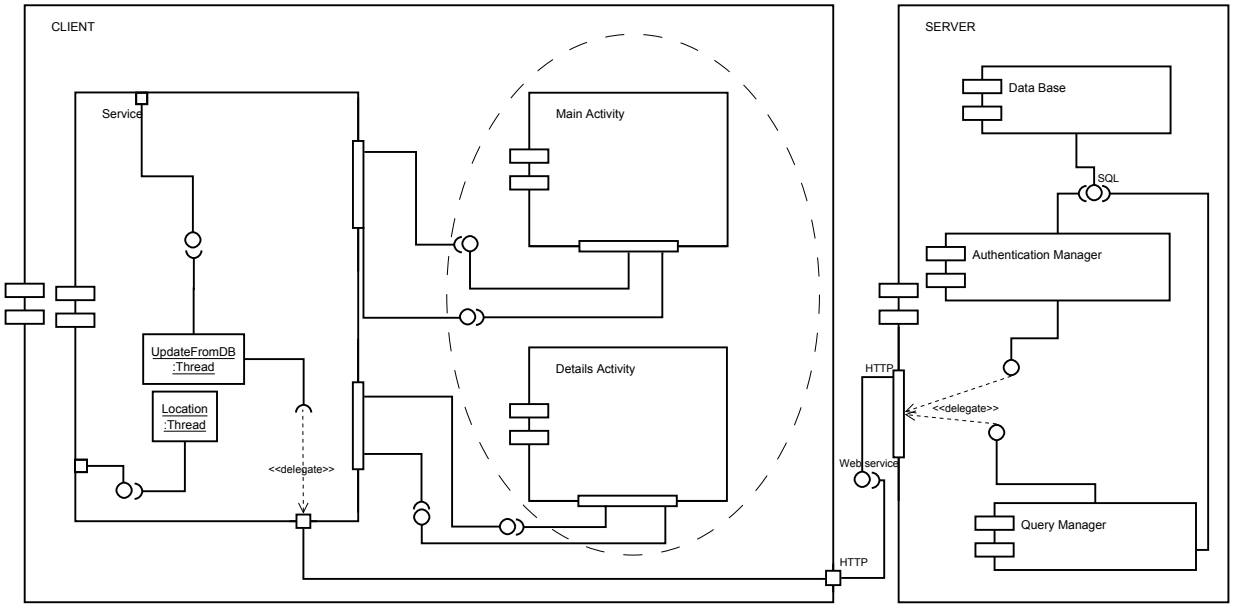


Figure 3: The components diagram of the client-server architecture, in the UML standard.

3.1 Android Client

We choose to build our client application on the Android platform. In fact, the Google mobile OS holds a large smartphones market share [25]. A typical Android application is composed of activities and services. An activity is the basic application component that provides a view for the user interface, while a service is an application component that does not provide any user interface, but performs background tasks.

Our Android application stands on three key components, as shown in Figure 3: two activities (main and details) and a service. A welcome activity and a settings activity were also developed in order to provide a graceful application launch and a reliable authentication and preferences handling.

The main activity shows the object list and let the user perform an update from the server. It is the most important application view and it also contains a menu entry for the settings activity.

The details activity shows a single object view, when the user wants to get a more accurate information on the object itself.

All the data are provided by an Android service, which is the most complex component; this service runs separate threads to handle location listening (for GPS or network coordinates retrieval) and to query the provider side. Every single remote request is performed by a dedicated thread. That way, we increase the user interface responsiveness, since each task runs asynchronously. The location thread runs for all the application life cycle and controls two listeners, one for GPS signals and one for network-based location information. Each listener is set to search for new coordinates every 30 seconds; when new coordinates are available, the thread chooses to send or not to send them to the service handler (Figure 4). If the service is not geo-located the new coordinates will always be sent, if the service is already geo-located, the more accurate GPS coordinates will be preferred. If the location thread does not receive new coordinates updates within a certain time interval, it sends the service handler a notification of lost signal. When the signal is lost the user can not update the object list but can ask for displayed object details.

The remote connection thread controls data flow from and to the provider side. It prepares the requests retrieving information from the application settings (such as username, password, bounding box and so on) and send them via an HTTP client. The parameter list is provided in Table 2.

The server side will provide a JavaScript Object Notation (JSON) formatted response containing the results. Some error responses such as *username not valid* or *password not matching user* are provided too. In order to handle lighter data on the mobile device, when asking for an object list a few data fields are considered (an object identifier, a name, a category and the object coordinates); these values are used to build a `LightObject` array that will inflate the Android ListView. The entire object data will be returned from the server only when the client asks for a single object details.

The Android service is connected to the application activities through Android Messengers (designed as double-sided interfaces). The service internal threads communicate via a single-side Android Handler interface instead.

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718-1876 Electronic Version
VOL 7 / ISSUE 3 / DECEMBER 2012 / 74-87

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762012000300008

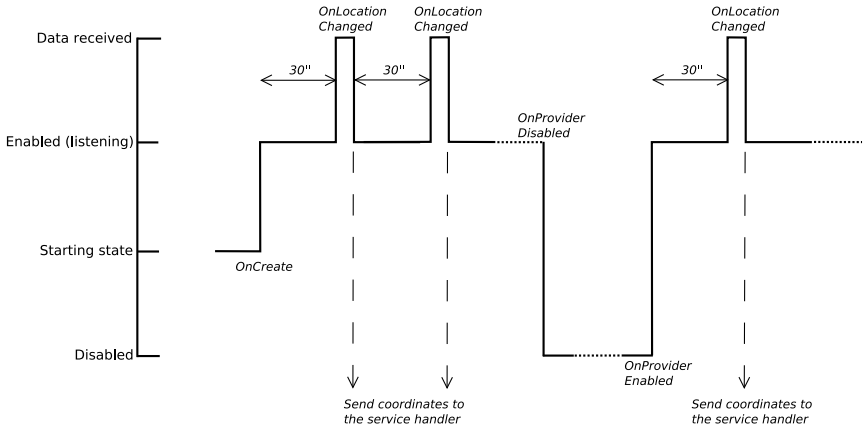


Figure 4: The timing logic behind the location thread.

Table 2: The parameters sent from Android clients to the provided http service in order to perform a remote request.

Field	Description	Values details
`username`	the username	String
`password`	the SHA-512 hash of the password	String
`job_id`	the remote request type	1: object list, 2: single object, 3: category list
`object_id`	the single object identifier	Int (0: no single object requested)
`filter`	the object categories we want to consider	Int[] (0: every category is considered)
`min_lon`	the bounding box minimum longitude	Float(10,6)
`max_lon`	the bounding box maximum longitude	Float(10,6)
`min_lat`	the bounding box minimum latitude	Float(10,6)
`max_lat`	the bounding box maximum latitude	Float(10,6)

3.2 Mirroring & Cloud

A key requirement of our infrastructure is scalability. The potential user base goes from the inhabitants of a single city, to those of the whole world. While in the former case a local server infrastructure might be sufficient to handle the requests from all the clients, in the latter a distributed solution is needed. Available technology, such as cloud computing and the mirroring and clustering capabilities of MySQL, makes this possible [20]. In particular, we want to achieve three goals. First, all clients should be able to query a server that is not physically distant, in order to decrease response time and enhance the user experience. Second, we want data replication (mirroring), that ensures a higher fault tolerance. Third, we want to have a single, integrated infrastructure, with centralized management, instead of a number of local, disconnected entities.

In our test case, we decided to use one of the available cloud computing services, Amazon Elastic Compute Cloud (Amazon EC2) (Site 1). EC2 is a web service that provides resizable computing capacity in the cloud. In practice, it offers an integrated computing service from data centres all over the world, which are collectively referred to as the *cloud*. EC2 servers are geographically distributed in many different regions (East and West US, EU, Asia Pacific and South America), which allows us to achieve goal number one, without compromising goal number three.

In order to have a mirrored and centrally controlled infrastructure (goals two and three), we use MySQL Cluster. MySQL Cluster is an open source transactional database. It is designed to be a distributed, multi-master architecture without single points of failure. In particular, our application can take advantage of the possibility to scale horizontally on different servers. Another key feature of MySQL cluster for our application is its ability to perform automatic data partitioning (sharding) with load balancing and the possibility to add nodes to a running cluster, that allows a great amount of scalability.

To explain how our construction can benefit from the use of Amazon EC2 and MySQL Cluster, we discuss an example of a simple distributed infrastructure, shown in Figure 5.

We have a three-levels scheme, where different nodes (servers) play different roles. Data nodes are where the information is stored; application nodes are where the web interface resides, get queried by the clients and act as an interface between them and the data nodes; and management nodes, used for the management and monitoring of the cluster.

In the management layer, the cluster configuration is stored on a set of management nodes. The configuration specifies node roles (management, application or data), their unique id and the tcp/ip connection rules between the nodes. The maximum number of nodes for the reference version of MySQL Cluster is 255.

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718-1876 Electronic Version
VOL 7 / ISSUE 3 / DECEMBER 2012 / 74-87

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762012000300008

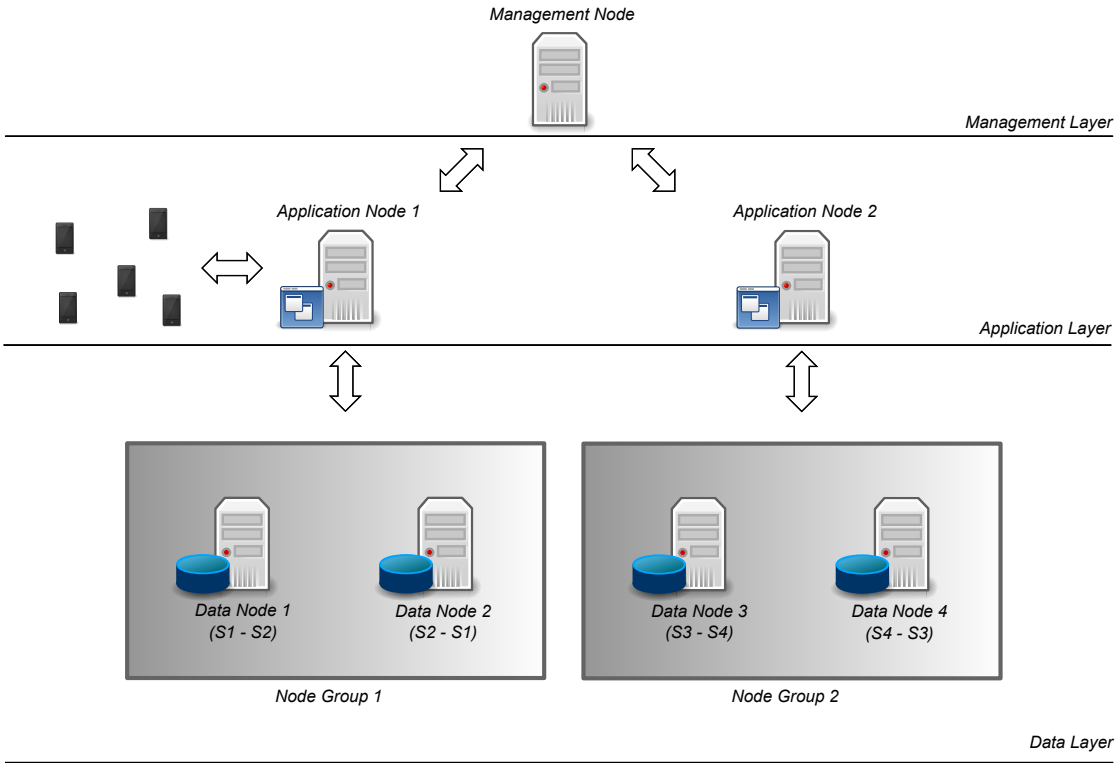


Figure 5: The provider side infrastructure in a distributed context, with data mirroring and partioning.

Application nodes expose the web service that serves client requests, and is composed of a set of geographically distributed servers. The client application has a built in list of a subset of these servers, called *entry nodes* (updates of the list are managed through the automatic updates of the application). Each entry server is assigned to a wide geographic area, but all of them can handle requests regardless of the client position. When the client needs to retrieve information, it tries to connect to the entry server associated with its current position. During this first request, the entry server can suggest a different server to use for subsequent queries, favoring servers that are geographically closer or subject to a lighter workload. In case of failure of any node, the client will temporarily exclude it from the list and interrogate a different entry server. All this happens transparently to the user.

On the data nodes, the most important information is stored in the `objects` table. To ensure the safety of the data, we replicate the information on different nodes. We also divide the nodes into groups, selected according to their geographical location. MySQL Cluster provides an *auto-sharding* function that let us divide data between data nodes. Usually a primary key hash criteria is used to balance the workload among data nodes, while in our case we prefer a *sharding* (division) based on the longitude and latitude fields, in order to obtain a range partitioning where each data node contains a cluster of geographically-related records.

Let us say, for example, that our application serves citizens of two different cities, one in Canada and one in South Korea. In this case we will have data nodes dislocated in both regions, and divided into two different groups, based on their proximity. Each group will store only local information, that is, information regarding the area they serve. At the group level, the `objects` table is further partitioned a number of times equal to the number of nodes within a group. We call this shares of information $S1$, $S2$ for Canada and $S3$, $S4$ for Korea (we suppose, for simplicity, that each group is composed of two nodes only). Each data node stores all the information of the data group (to ensure data replication), but for each partition S only one node will act as *master*, meaning that it will be responsible of data management. The other nodes will act as *slaves*. So, in our example, the first node of the Canadian group will store $S1$ as master and $S2$ as slave, while the second will store $S1$ as slave and $S2$ as master. In case of failure, the group is able to continue operating as long as at least one node is still running.

One application node in Canada will receive the queries from local mobile users, and will request the needed information to the local data group. The same happens in Korea. In the case of an application node becoming unavailable, the requests are dynamically and transparently redirected to the other node. This will have an impact on the responsiveness, but the service will still be fully functioning. All the management and monitoring is done from the dedicated management node. The geographical position of this node is irrelevant to the functionality of the system.

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718-1876 Electronic Version
VOL 7 / ISSUE 3 / DECEMBER 2012 / 74-87

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762012000300008

3.3 Database

Since we want to keep queries as simple as possible, and avoid unnecessary `JOIN` operations, we keep the database structure quite simple. It includes some tables serving the authentication procedure (stored with the InnoDB engine in order to support transactions), the table `object` that contains objects data and the table `category` that contains object's categories. These two tables are stored with the MyISAM storage engine to improve query responsiveness.

In the distributed context described in Section 3.2 tables are stored with the NDBCluster storage engine; query responsiveness still stands (principally due to the fact that NDBCluster loads every data entry in the server memory) as transaction support, even if limited to *read committed* isolation level [26].

The large amount of fields of the table `object` is due to the fact that it stores objects of all the different types. We query this table for a small amount of fields when asking for an objects list (the complex scenario), and for all the table fields when asking for a single object by id (the simple scenario). That way, a list job will not produce a massive response to the client.

Let us focus on tables join (upon the ref key `categotyId` that links objects with their categories) in the complex object list scenario: we want to avoid any kind of join between the two tables to speed up our queries. This is accomplished thanks to the class-factory included in the Android client: this class knows the logic of the categories database identifiers and can map every retrieved object to the correct category directly on the client mobile device, using the `categoryId` value included in table `object`.

One of the distinguishing traits of a smart city is the exposure of on-demand intensive services to the urban actors. This usually implies a dedicated On-Line Transaction Processing (OLTP) distributed infrastructure that can ensure the highest possible reliability. This is usually implemented using the concept of DataBase as a Service (DBaaS) [4]. In this paper, we address this problem with a new, integrated approach based on cloud computing and database clustering. Since mobile services in the urban context often need some form of geo-location of the clients, in order to provide location-aware services, a common research topic is efficient processing of spatial information. In [17], Mondal et al. propose a new database index structure, designed to improve the efficiency of spatial queries. We propose an original solution to the problem in Section 3.3.3.

3.3.1 Interface to the DB

In order to avoid the need of a direct connection to the DB, which may not be possible on mobile Internet connections, we provide on the server side an HTTP interface. The data are transferred using JSON objects, that are generated on the fly through a PHP script on the server. The communication happens through HTTP POST requests, one of the standard request mechanisms specified by the HyperText Transfer Protocol (HTTP). An object list request performed by the client will result in the server outputting a JSON formatted string containing the values shown in Table 3. These values are then used to fill the object list that appears in the user interface of the Android client.

Table 3: The parameters sent from the server to the client after an object list request. Data are Json formatted.

Field	Description	Values details
`id`	the object identifier	BigInt(20), unsigned
`name`	an object description	Varchar(60)
`categoryId`	the object's category identifier	Int(10), unsigned
`longitude`	the object's longitude	Float(10,6)
`latitude`	the object's latitude	Float(10,6)

3.3.2 Range Query Optimization

In order to obtain a powerful OLTP architecture we want to minimize the execution time for the object list retrieval query. Instead of including real distance evaluation within the query itself, as proposed by [15], we perform a simple range query (i.e. a square window query). That way, we consider a larger area around the citizen (approximately 21.46% wider), but we keep the query simple and we minimize its execution time. In fact, we do not compute any distance function inside the query itself. Instead, we just specify the area through two `BETWEEN` statements in the `SQL WHERE` clause.

Objects falling into the exceeding area will be discarded by the Android client after an exact evaluation of their relative distance from the user. The client itself will also be responsible of bounding box evaluation and objects sorting. These concepts are shown in Figure 6: first the client computes the bounding box; then the database server performs the range query; finally the client is responsible of tasks like object sorting and real distance evaluation. The offloading of part of the computation from the DB server to the mobile client let us reduce the workload of the DBMS, while maintaining a precise calculation of the desired area. The improvement we achieved in the execution time of the query is on average 23.5%. Scaling up the time of a single execution to a number of queries that can reach thousands per minute, the benefits of this approach are evident.

The bounding box is computed using the Java GeoLocation class provided in [15], in order to avoid possible miscalculations in the regions near the North and South geographic poles and surrounding the 180th Meridian.

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718-1876 Electronic Version
VOL 7 / ISSUE 3 / DECEMBER 2012 / 74-87

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762012000300008

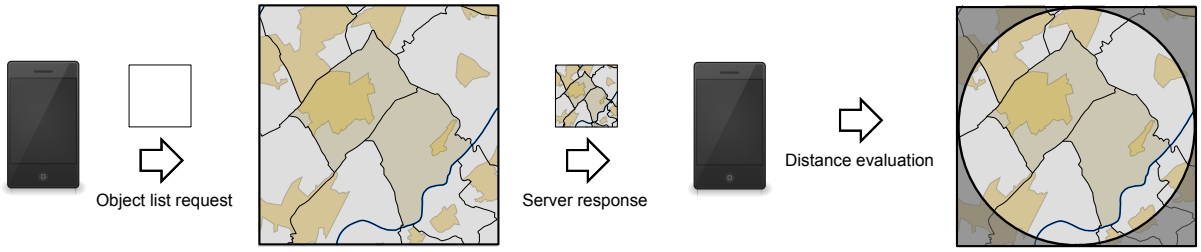


Figure 6: The process behind an object list remote query.

3.3.3 Indices

A requisite of primary importance for a responsive OLTP architecture is a fast execution of SQL queries. We already described how pre-processing and post-processing performed on the mobile client can ensure a lighter computational load on the server and a simpler query syntax. However, the most important implementation choice is certainly the kind of indices for database tables. In the described context the database will be continuously queried for objects in small square regions. This kind of query, called *range query*, is notoriously well served by B+-trees [3] or, in the case of a geometrical context, by R-trees [8]. As we consider geo-located objects, we test both R-trees and B+-trees, in order to select the most performing option for our application. Since MySQL does not natively support the more advanced kNR-trees [17], in the first case we consider a simple R-tree, and access specific categories using the SQL `WHERE` clause.

In our tests we considered both stand alone and distributed MySQL servers. The most computationally intensive queries are performed on the `object` table. In particular, we want to index the spatial information: the `longitude` and `latitude` fields, for B+-trees, as well as the `point` field in the case of R-trees. The `point` field is a MySQL representation of a geometric point (containing one longitude and one latitude value), belonging to the `geometry` class. In fact, MySQL supports R-tree indices only for `geometry` data.

MySQL uses R-Trees with quadratic splitting for `SPATIAL` indices on spatial columns. A `SPATIAL` index is built using the *Minimum Bounding Rectangle* (MBR) of a geometry [26]. For most geometries, the MBR is the smallest rectangle that surrounds the geometry. For a point, the MBR is a rectangle degenerated into the point. For tables stored with MyISAM storage engine, `SPATIAL INDEX` creates an R-tree index. It is also possible to index a spatial column with a B-tree. Nevertheless, a B-tree index on spatial values will be useful for exact-value lookups, but not for range scans. That is why we use B+-trees on `longitude` and `latitude` fields (that are stored in `float(10,6)` format) and a R-tree on `point` field (that is stored in a MySQL `POINT` data structure). The integrated query optimizer uses available spatial indices every time functions such as `MBRContains()` or `MBRWithin()` are present in the `WHERE` clause of the query.

The two following SQL queries reflect the statements that were actually used in our experiment, the first one for B+-tree indices and the second one for R-tree indices. These two queries are syntactically different but they have the same semantic meaning, i.e. they query the data base for objects in the same bounding box.

```
SELECT id, name, idCategory, longitude, latitude FROM object
WHERE latitude BETWEEN x1 AND x2 AND longitude BETWEEN y1 AND y2;

SET @boundingBox = 'Polygon((x1 y1, x2 y1, x2 y2, x1 y2, x1 y1))';
SELECT id, name, idCategory, AsText(point) FROM object
WHERE MBRContains(GeomFromText(@boundingBox),point);
```

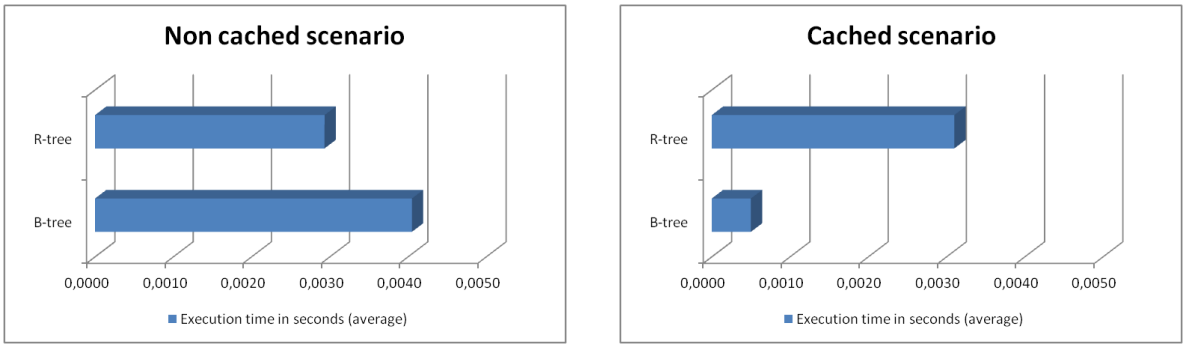
For each kind of index, we performed multiple tests, each selecting areas with a different number of objects. We further expanded the experiment first by evaluating additional queries for which a single object category is retrieved, and then by enabling and disabling the caching of the queries.

Query caching is an important factor for the performances of an OLTP architecture. Test results on a realistic instance of a sample geographical dataset in Figure 7 show a remarkable difference between cached and non cached queries. In the non cached scenario, R-trees are approximately 30% faster than B+-trees, while in the cached scenario B+-trees are roughly 84% faster than R-trees. These tests were performed on a single server. In the distributed scenario we considered in Section 3.2 we can not use the MyISAM storage engine, since a MySQL cluster requires the NDBCluster storage engine. Since NDBCluster does not support `SPATIAL` indices, in a distributed context we have to use B+-trees.

It is remarkable that the best way to obtain a scalable and high performance OLTP architecture consists in using NDBCluster tables indexed with B+-trees and a wide usage of query cache. However, a query can be recalled by MySQL from the cache only if it is exactly identical to the current one. The comparison is a real byte to byte equality check, so, for instance, `SELECT * from object` is different from `select * from object`. Since even two client devices located only meters one from another would produce queries with different longitude and latitude parameters, query caching can not be effectively used. A solution to this limitation could be to pre-process the bounding box on the server and then match the queries to already cached queries from devices within a certain distance, using for example some hash criteria. These issues surely represent an interesting subject for future research.

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718-1876 Electronic Version
VOL 7 / ISSUE 3 / DECEMBER 2012 / 74-87

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762012000300008



(a) Non cached

(b) Cached

Figure 7: Indices tests: R-trees perform better in a non-cached context while B+-trees do in the cached one.

3.4 Security

Security is a crucial factor for the success of a complex infrastructure like the one we present in this paper. Moreover, wherever user interaction is required, and user accounts are created, privacy measures need to be taken, according to current legislation. In this section we describe how we secure key components of the system, and what precautions we employ in order to guarantee an adequate level of privacy to the perspective users of the service.

User authentication is one of the most critical components from a security point of view. We use cryptographic techniques for password transmission, and design the HTTP authentication mechanism with SQL injection resistance in mind. Direct SQL Command Injection is a technique where an attacker tries to alter existing SQL commands to modify or get access to hidden data. If no precautions are taken, an attacker can even execute dangerous system level commands on the database host. This is accomplished through the use of badly formatted user input given to the application. This malicious inputs usually use static parameters to build an SQL query.

As suggested in [16], we used a form of stack guard that prevents our database from being SQL injected. First of all, every parameter read from the HTTP POST request (shown in table 2) is truncated to the maximum sensible length and passed to a function that escapes special characters, taking into account the current character set of the data base connection so that it is safe to be placed in a query. Moreover, each query executed through PHP within the server HTTP interface (3.3.1) runs under the MySQL Improved Extension (mysqli). We only use prepared statements for query execution, following the *prepare - bind - execute* pattern; that way a statement template is send to the database server, the server performs a syntax check and initializes internal resources for later use. The client can bind parameter values and sends them to the server. The server creates a statement from the statement template and the bound values to execute it using the previously created internal resources.

For authentication purposes, we store and transmit only the hash of the password. The hash function we choose is SHA-512, part of the SHA-2 family, and currently the strongest function approved by the United States National Institute of Standards and Technology (NIST) [11].

As proposed in [4], additional security checks should be performed when sensitive data are stored in the cloud. In particular, Curino et al. propose the use of an enhanced security scheme that allows SQL queries to be run over encrypted data, including ordering operations, aggregates and joins. As cloud services are expected to sustain a strong growth in the near future, and are candidate to partially replace in-house computing solutions, the issue surely represent an interesting research topic for future work.

4 System Adoption and Future Research

The system we describe in this paper is currently being implemented within a collaboration between the *Smart City Lab* (Site 2) of the Università di Bologna and the Italian municipality hosting the campus, Cesena. This experimental implementation follows the general scheme proposed in this work, but is currently limited in the infrastructure to a centralized database solution due to the limited number of items and users that will be served in this first experiment. The application, currently in an advanced stage of development, will be distributed to the public for free through Android official application market (Google Play). Appendix 5 contains some screenshots of the latest version at the time of writing.

The municipality of Cesena has provided support to the project by allocating human resources and contributing to define the taxonomy of the points of interest included. After a survey of public preferences, the set of categories included in the first release has been fixed to the following: museums, hotels, restaurants, gas stations, supermarkets, bookstores, cinemas, clubs, hospitals and pharmacies. The university took care of the software development and all the technical aspects of the project. Prior to the public release of the application, the software will be further enhanced by introducing an interface that will let the user suggest new items or improvements to the existing ones.

A future expansion of this first experiment, or an implementation done by a private company, could involve a dedicated

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762012000300008

database as a service (DBaaS) infrastructure managed by an independent party (for instance, a company specializing in data management). In this scenario, the service provider would only be responsible of the development of the application and its marketing, delegating the infrastructure component to a third party.

One of the possible sources of revenue for this kind of project is advertisement. The main idea behind the well-known Google advertising program is that advertisers appear on the first page of the search result, regardless of their page rank. We can apply the same philosophy to our application. For example, a gas station company or any other business actor could pay an advertising fee in order to appear in the topmost slot of the object list.

Contrary to what happens in a search engine, where the advertising results are weakly location-dependent (in Google, for instance, sending location information is optional, and in the case this information is not provided the location of the user is guessed using the IP address), here we return a location-aware object list. So, advertising is location-aware too. This could boost the interest in this type of advertising: in its own neighbourhood, an advertiser has the possibility to be the first result despite the system receiving higher payments from companies elsewhere. The advertisers could also be offered premium memberships, that would allow them to have custom object classes (for instance, having the company logo as icon) or customized detail views, that allow the advertisers to describe their objects in a personalized way.

Due to the mobile nature of the system, wireless carriers or mobile network operators could take part in the project, either by contributing to its dissemination or even by taking the role of service provider.

5 Conclusion

In this paper we present a location based, client-server service designed to retrieve useful objects nearby the citizen. The users of the service use a client application on mobile devices, such as tablets and smartphones; the provider of the service relies instead on a complex On-Line Transaction Processing (OLTP) server infrastructure, designed to execute spatial queries over a wide amount of data.

While the application we present as a case study is innovative by itself, comprehensive studies on the entire production cycle of location-aware smart city applications and the related issues are still missing from scientific literature. This paper fills that void, providing a study on the design, implementation and deployment of a smart mobile application that retrieves, and conveys to the user relevant information coming from a city's intelligent infrastructure.

In particular, we discuss location-aware and mobile development, spatial data storage, infrastructure clustering and security, all common problems in the design of smart cities applications. We also analyze currently available technology, and show how it can be used to implement the service in an efficient and effective way. We also introduce a novel technique to perform range queries upon spatial data, that stands on the assumption that the client can improve the object retrieval process with some pre- and post- data processing. Finally, we provide the results of experimental testing on the efficiency of different data structures for database indices of spatial data tables.

In this work we have addressed many significant issues in the development of smart applications for smart cities. The framework we provided is at the basis of the city intelligent infrastructure, and will allow the design of smarter applications for better cities. The Italian city of Cesena, in collaboration with the Università di Bologna, has adopted this solution for its citizens and is currently sponsoring its implementation.

As smart cities are a relatively new and developing concept, our contribution opens the way to future works on the subject, and we believe that we contributed to prepare a solid ground over which to build the cities of tomorrow.

Acknowledgments

The research work presented in this paper has been coordinated by the Smart City Lab, at the Università di Bologna. Paolo Palmieri is supported by the SCOOP Action de Recherche Concertées.

Websites List

Site 1: Amazon Elastic Compute Cloud (Amazon EC2)
http://aws.amazon.com/ec2/

Site 2: Smart City Lab (Università di Bologna, Italy)
http://smartcity.csr.unibo.it/

References

[1] P. Borokhov, S. Blandin, S. Samaranayake, O. Goldschmidt, and A. Bayen, An adaptive routing system for location-aware mobile devices on the road network, in Proceedings of the 14th International IEEE Conference on Intelligent Transportation Systems, 2011, pp. 1839-1845.

[2] F. Calabrese, K. Kloeckl, and C. Ratti, WikiCity: Real-Time Location-Sensitive Tools for the City, Digital Cities 5: Urban Informatics, Locative Media and Mobile Technology in Inner-City Developments, Workshop, 2007.

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762012000300008

[3] D. Comer, The Ubiquitous B-Tree, ACM Computing Surveys, vol. 11, no. 2, pp. 121-137, 1979.
[4] C. Curino, E. P. C. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan, and N. Zeldovich, Relational Cloud: A Database Service for the Cloud, in Proceedings the 5th Biennial Conference on Innovative Data Systems Research, Pacific Grove, 2011, pp. 235-240.
[5] J. Domingue, A. Galis, A. Gavras, T. Zahariadis, D. Lambert, F. Cleary, P. Daras, S. Krco, H. Müller, M.-S. Li, H. Schaffers, V. Lotz, F. Alvarez, B. Stiller, S. Karnouskos, S. Avessta, and M. Nilsson, The Future Internet - Future Internet Assembly 2011: Achievements and Technological Promises, Lecture Notes in Computer Science, vol. 6656, Berlin: Springer-Verlag, 2011.
[6] European Commission, Advancing and Applying Living Lab Methodologies, Technical Report, 2010.
[7] R. Giffinger, C. Fertner, H. Kramar, R. Kalasek, N. Pichler-Milanović, and E. Meijers, Smart Cities: Ranking of European Medium-sized Cities, Centre of Regional Science (SRF), Vienna, Technical Report, 2007.
[8] A. Guttman, R-Trees: A Dynamic Index Structure for Spatial Searching, in Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, ACM, New York, 1984, pp. 47-57.
[9] O. Haubensak, Smart Cities and Internet of Things, Business Aspects of the Internet of Things, Seminar of Advanced Topics, ETH Zurich, 2011, pp. 33-39.
[10] J. M. Hernández-Muñoz, J. B. Vercher, L. Muñoz, J. A. Galache, M. Presser, L. A. H. Gómez, and J. Pettersson, Smart Cities at the Forefront of the Future Internet, in The Future Internet (J. Domingue, A. Galis, A. Gavras, T. Zahariadis, D. Lambert, F. Cleary, P. Daras, S. Krco, H. Müller, M.-S. Li, H. Schaffers, V. Lotz, F. Alvarez, B. Stiller, S. Karnouskos, S. Avessta, and M. Nilsson, Eds.). Heidelberg, Berlin: Springer-Verlag, 2011, pp. 447-462.
[11] Information Technology Laboratory and NIST, Secure hash standard (SHS), U.S. Dept. of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, 2008.
[12] International Telecommunication Union, The Internet of Things, ITU Internet Report, 2005.
[13] ITU-T TSAG, A Preliminary Study on the Ubiquitous Sensor Network, Technical report, 2007.
[14] C. Klein and G. Kaefer, From Smart Homes to Smart Cities: Opportunities and Challenges from an Industrial Perspective, in Proceedings of the 8th International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking, St.-Petersburg, 2008, pp. 260.
[15] J. P. Matuscheck, Finding Points Within a Distance of a Latitude/Longitude Using Bounding Coordinates, Technical Report, 2011.
[16] E. Merlo, D. Letarte, and G. Antoniol, Automated Protection of PHP Applications Against SQL-injection Attacks, in Proceedings of the 11th European Conference on Software Maintenance and Reengineering, IEEE Computer Society, Washington, 2007, pp. 191-202.
[17] A. Mondal, A. K. H. Tung, and M. Kitsuregawa, kNR-tree: a novel R-tree-based index for facilitating spatial window queries on any k relations among N spatial relations in mobile environments, in Proceedings of the 6th International Conference on Mobile Data Management, ACM, New York, 2005, pp. 173-177.
[18] T. Nam and T. A. Pardo, Conceptualizing smart city with dimensions of technology, people, and institutions, in Proceedings of the 12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times, New York, 2011, pp. 282-291.
[19] T. Nam and T. A. Pardo, Smart city as urban innovation: focusing on management, policy, and context, in Proceedings of the 5th International Conference on Theory and Practice of Electronic Governance, ACM, New York, 2011, pp. 185-194.
[20] Oracle Corporation, Guide to Scaling Web Databases with MySQL Cluster, Technical Report, 2012.
[21] J.-W. Park, C. H. Yun, S. W. Rho, Y. Lee, and H.-S. Jung, Mobile Cloud Web-Service for U-City, in Proceedings of 9th International Conference on Dependable, Autonomic and Secure Computing, Sydney, 2011, pp. 1061-1065.
[22] F. M. Rojas, K. Kloeckl, and C. Ratti, Dynamic City: Investigations into the sensing, analysis, and applications of real-time, location-based data, Proceedings of Harvard GSD Critical Digital Conference, Cambridge, 2008.
[23] H. Schaffers, N. Komninos, and M. Pallot, Smart Cities as Innovation Ecosystems Sustained by the Future Internet, FIREBALL, Technical Report, 2012.
[24] H. Schaffers, N. Komninos, M. Pallot, B. Trousse, M. Nilsson, and A. Oliveira, Smart Cities and the Future Internet: Towards Cooperation Frameworks for Open Innovation, in The Future Internet (J. Domingue, A. Galis, A. Gavras, T. Zahariadis, D. Lambert, F. Cleary, P. Daras, S. Krco, H. Müller, M.-S. Li, H. Schaffers, V. Lotz, F. Alvarez, B. Stiller, S. Karnouskos, S. Avessta, and M. Nilsson, Eds.). Heidelberg, Berlin: Springer-Verlag, 2011, pp. 431-446.
[25] The Nielsen Company, Two Thirds of New Mobile Buyers Now Opting For Smartphones, Technical Report, 2012.
[26] M. Widenius and D. Axmark, Mysql Reference Manual, 1st ed., CA: O'Reilly & Associates, Inc., 2002.
[27] L. Zhang, S. Gupta, J.-Q. Li, K. Zhou, and W. bin Zhang, Path2Go: Context-aware services for mobile real-time multimodal traveler information, of Proceedings of the 14th International IEEE Conference on Intelligent Transportation Systems, Washington, 2011, pp. 174-179.

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718-1876 Electronic Version
VOL 7 / ISSUE 3 / DECEMBER 2012 / 74-87

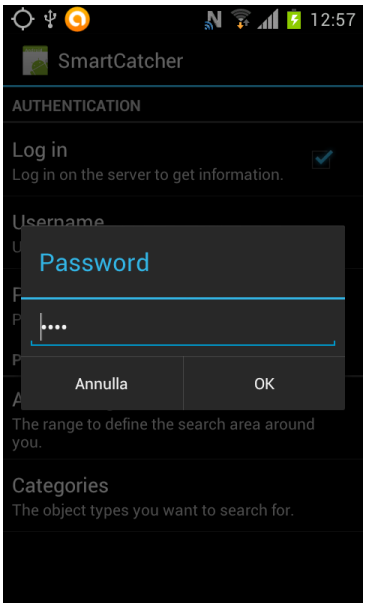
This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762012000300008

Appendix A: Application Screenshots

Screenshot (a) shows a list of interesting objects near the user. The distance from the current position is displayed on the right, while the icon on the left depends on the category of the object. Information on the second object in the list, displayed by the details activity, are shown in screenshot (b). In picture (c) the user is setting the password, from the authentication option of the *settings* menu.

(a) Main view

(b) Object information



(c) Preferences